**PRIMERA PARTE**

1.

Crear books.html

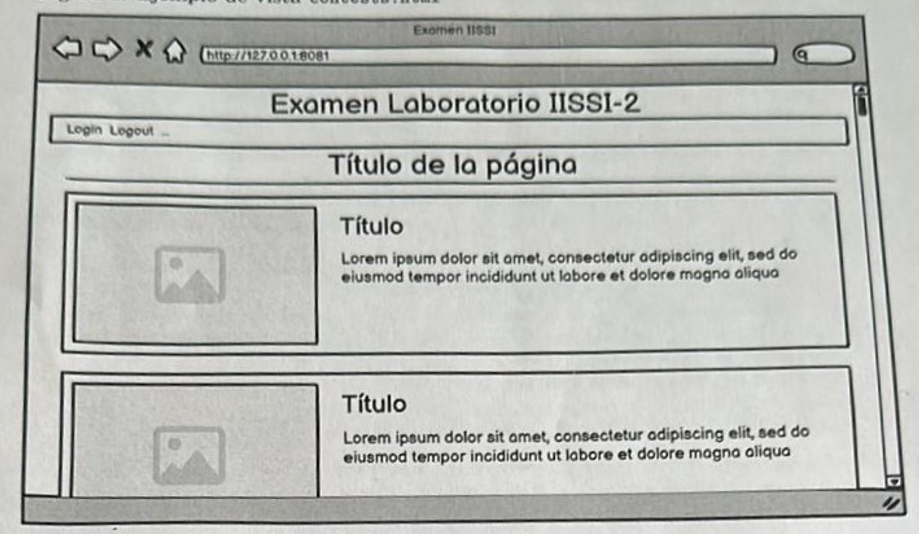
Copiar y pegar cabeceras de min index.html

Copiar de container a section de otro examen.

En htmls cambiar solo título y nombres de archivos.

2.

Crear booksRenderers.js (copiando de otro examen) y modificar atributos/formato (nombres igual que en Heidi)



let html = `<div class="card flex-row">

                        <img class="card-img w-25 p-2" src="${contest.imageUrl}" alt="${contest.name}">

                        <div class="card-body col">

                            <h3><b>${contest.name}</b></h3>

                            <p><b>Fecha: </b>${contest.contestDate}</p>

                            <p><b>Precio: </b>${contest.prize}</p>

                            <p><b>Descripción: </b>${contest.description}</p>

                        </div>

                    </div>`

Diagrama

Descripción generada automáticamente

asCard: function (book) {

        let html = `<div class="col-sm-4 card mb-1 p-3 ">

        <div class="">

            <a>

<!-- <a href="artists\_details.html?artistId=${artist.artistId}" class="text-muted"> Para referenciar un enlace con la info -->

                <img src="${book.imageUrl}" class="img-fluid w-100" >

<!-- style="width: 340px;height: 250px Lo añadimos al final para que todas tengan el mismo tamaño -->

            </a>

        </div>

        <div class="card-body text-center m-0 p-3">

            <h5 class="card-title m-0">#${book.bookId} ${book.title}</h5>

            <p class="card-text m-0 ">${book.author}</p>

            <!-- BOTON -->

            <br>

            <a href="create\_books.html?bookId=${book.bookId}" ><button  class="btn btn-secondary col mx-auto">Editar</button></a>

        </div>

    </div>`;

3.

Crear galleryRenderer.js (copiado de otro examen) -> Contenedor de cartas

"use strict";

import { parseHTML } from "/js/utils/parseHTML.js";

import{contestsRenderer} from "./contestsRenderers.js";

const galleryrenderer = {

    asCardGallery: function (contests) {

        let galleryContainer = parseHTML('<div class="contest-gallery row p-5 bg-light"> </div>');

        for (let contest of contests) {

            galleryContainer.appendChild(contestsRenderer.asCard(contest));

        }

        return galleryContainer;

    }

};

export { galleryrenderer }

4.

Contest.js (fuera de renderers). Siempre igual.

En gallery metemos carta (un libro) y aquí metemos la galeria

"use strict";

import { galleryrenderer } from "./renderers/gallery.js";

import {messageRenderer} from "./renderers/messages.js";

import {booksAPI\_auto} from "./api/\_books.js";

async function main() {

    try {

        let galleryBooks = await booksAPI\_auto.getAll();

        let divGallery = document.getElementById("divGallery");

         //este "divGallery"es el de book.html

        divGallery.appendChild(galleryrenderer.asCardGallery(galleryBooks));

    }

   catch(err){

    messageRenderer.showErrorMessage('Error al crear el libro: ' + err.response.data.message);

  }

}

document.addEventListener("DOMContentLoaded", main);

**SEGUNDA PARTE (FORMULARIO)**

1.

Crear create\_books.html y hacer formulario (copiar y pegar cambiando los parámetros y puede cambiar lo de columnas/filas)

Copiar cabecera-aside [METER FORMULARIO] section-script

Crear create\_books.js (copiar y pegar)

"use strict";

import {messageRenderer} from "./renderers/messages.js";

import { booksAPI\_auto } from "./api/\_books.js";

async function main() {

    alert("create\_books.js controlador enlazado");

    document.getElementById("formId").onsubmit= async function (event){

        let formData = new FormData(event.target);

        event.preventDefault();

                sendRegister(formData);

    }}

async function sendRegister(formData){

    try {

        await booksAPI\_auto.create(formData);

    } catch (err) {

        messageRenderer.showErrorMessage('Error al crear el libro: ' + err.response.data.message);

    }

}

document.addEventListener("DOMContentLoaded", main);

Para formato de email válido: (placeHolder es lo que sale en el recuadro)

 <div class="form-group">

                    <label for="inputData"> Email:</label>

                    <input type="text" class="form-control" id="inputData" name="inputData"

                        placeholder="ejemplo@dominio.com" pattern="^[a-zA-Z0-9]+@[us.es]" required>

                    <div class="invalid-feedback">

                        Por favor, introduce un dato válido (ejemplo: texto@us.es).

                    </div>

                </div>

**TERCERA PARTE (BOOKS)**

Editar, añadir, eliminar

En delete\_book.js aparece como hacer para que salga mensaje de confirmación y como registrar un usuario.

Botón editar libro en renderer porque es uno para cada libro/carta.

Restricción editar/crear libro en create\_book.js (VER EN EXAMEN LIBROS)

Si para poder editar/crear/eliminar tengo que estar logueado:

 if (!sessionManager.isLogged()) {

        alert ("No puede editar fotos si no está autorizado!");

        window.location.href = `books.html`; // Cancelar edición si no está autorizado

    }

**TERCERA PARTE (CONTEST)**

En header.html (li)

<li class="nav-item" id="header-contets">

                        <a class="nav-link" href="contests.html">

                            Contests

                        </a>

                    </li>

                    <li class="nav-item" id="header-create\_contest">

                        <a class="nav-link" href="create\_contest.html">

                            Create Contest

                        </a>

                    </li>

**OTROS**

Si da error d permisos y no me dicen q me loguee: endpoints api y en los dos json cambiar true por false y volver a hacer silence run

**TIPOS DE BOTONES**

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Para que se autocomplete con algún dato de la persona logueada

import { sessionManager } from "/js/utils/session.js";

async function loadCurrentUser() {

    let userId = document.getElementById('id-input');

    try {

        let loggedUser = sessionManager.getLoggedUser();

        userId.value =  loggedUser.username;

    }

    catch (err) {

        messageRenderer.showErrorMessage('Error al cargar el userId: ' + err.response.data.message);

    }

}

Para que se autocomplete con la fecha actual

async function loadCurrentDate() {

    let dateInput = document.getElementById('releaseDate-input'); // Obtiene el elemento de entrada para la fecha

    let today = new Date(); // Crea un nuevo objeto de fecha con la fecha y hora actuales

    let year = today.getFullYear(); // Obtiene el año actual (e.g., 2023)

    let month = String(today.getMonth() + 1).padStart(2, '0'); // Obtiene el mes actual (de 0 a 11) y le suma 1. Luego lo convierte a una cadena y agrega un cero inicial si es necesario (e.g., '09')

    let day = String(today.getDate()).padStart(2, '0'); // Obtiene el día del mes (de 1 a 31). Luego lo convierte a una cadena y agrega un cero inicial si es necesario (e.g., '05')

    let currentDate = `${year}-${month}-${day}`; // Formatea la fecha en el formato YYYY-MM-DD, requerido por el campo de entrada de tipo date

    dateInput.value = currentDate; // Establece el valor del campo de entrada de fecha con la fecha actual

}